强化学习

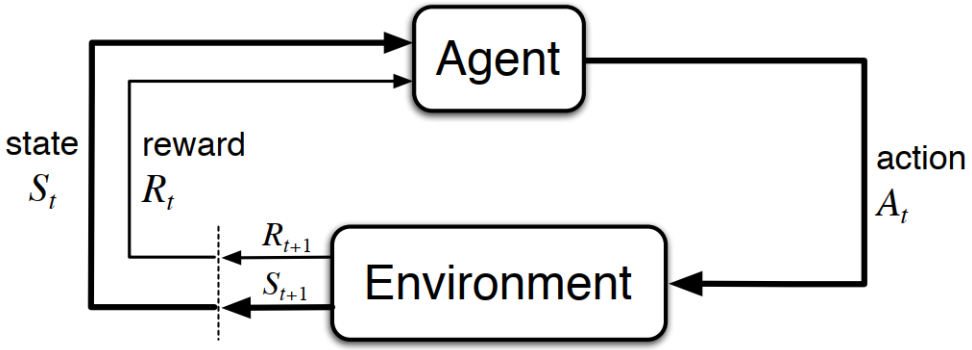
12月23日

姚鑫

- 简单介绍
- Temporal difference方法
- Policy gradient方法

Introduction

- 强化学习是机器学习中的一个领域，强调如何基于环境而行动，以取得最大化的预期利益。
  - 在运筹学和控制理论中，被称作“近似动态规划”。在最优控制理论中，主要关注最优解以用精确计算的算法，而不是学习或近似。
  - 在机器学习中，环境通常被表述为马尔可夫决策过程(MDP)。
- 强化学习系统
  - MDP下的智能体(agent)与环境(environment)



Introduction

- 强化学习系统
  - a policy, a reward signal, a value function, a model(optional)
    - ✓ a policy: 从状态到动作的映射。随机策略时， 可以表示为一个概率分布， $\pi : \mathcal{S} \times \mathcal{A} \to \mathcal{R}$
    - ✓ a reward signal: 一个关于状态和动作的实值函数，可以表示为，$R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathcal{R}$
    - ✓ a value function: 代表在给定状态或状态动作对下的累积奖励，分别称为state-value function，$V : \mathcal{S} \to \mathcal{R}$， state-action value function，$Q : \mathcal{S} \times \mathcal{A} \to \mathcal{R}$
    - ✓ A model: 允许推断环境的行为方式，比如，给定状态和动作来预测下一状态和奖励。

Markov Decision Processes

- 马尔可夫决策过程是一个四元组 $(\mathcal{S}, \mathcal{A}, P, R)$
  - $\mathcal{S}$: 有限的状态集合
  - $\mathcal{A}$: 有限的动作集合
  - $P(s'|s,a) = Pr(s_{t+1} = s'|s_t = s, a_t = a)$ 给定动作a时从状态s到状态s'的概率
  - $R(s,a,s')$: 得到的即时奖励
- 下一状态只由当前状态和动作决定，与之前的状态无关

Policies and Value Functions

- define the return

$$G_t \doteq \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$$

- define the state-value function

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s\right], \text{ for all } s \in \mathcal{S},$$

- define the action-value function

$$q_\pi(s,a) \doteq \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s, A_t = a\right]$$

- optimal policy

$$v_*(s) \doteq \max_\pi v_\pi(s)$$

$$q_*(s,a) \doteq \max_\pi q_\pi(s,a)$$

Bellman equation

- 贝尔曼方程
  - $V^{\pi}(s) = r + \gamma \mathbb{E}_{s' \sim P(s'|s,\pi)}[V^{\pi}(s')|s,\pi]$
  - $Q^{\pi}(s,a) = \mathbb{E}_{s' \sim P(s'|s,a)}[r + \gamma \mathbb{E}_{a' \sim \pi} Q^{\pi}(s',a')|s,a,\pi]$
- 求解方法
  - dynamic programming
    - ✓ a model of the environment
  - Monte Carlo methods
    - ✓ wait for the end of an episode
  - temporal difference learning
    - ✓ no model and fully incremental

求解方法

- 迭代形式

$$\begin{aligned} v_\pi(s) &\doteq \mathbb{E}_\pi[G_t \mid S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s] \end{aligned}$$

- dynamic programming
  - v_π 用 V近似，期望可求
- Monte Carlo methods
  - every-visit Monte Carlo method

$$V(S_t) \leftarrow V(S_t) + \alpha\Big[G_t - V(S_t)\Big]$$

  - G_t 可求，期望用采样近似
- TD
  - $V(S_t) \leftarrow V(S_t) + \alpha\Big[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\Big]$
  - v_π用 V近似，同时期望用采样近似

Algorithm

- on-policy or off-policy
  - π — target policy(被估计的策略)
  - μ — behavior policy(产生数据的策略，更好地探索)
  - $\pi = \mu$: on-policy; $\pi \neq \mu$: off-policy
  - off-policyness: $\varepsilon := \max_x ||\pi(\cdot|x) - \mu(\cdot|x)||_1$
- discrete actions or continuous actions
- prediction problem or control problem
- value iteration or policy iteration or policy gradient or evolution

- 简单介绍
- Temporal difference方法
- Policy gradient方法

Temporal difference

- one-step TD
  - the target of Monte Carlo update
    - ✓ $G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots + \gamma^{T-t-1} R_T$
  - the target in one-step return
    - ✓ $G_{t:t+1} \doteq R_{t+1} + \gamma V_t(S_{t+1})$
- n-step TD
  - the target in two-step return
    - ✓ $G_{t:t+2} \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 V_{t+1}(S_{t+2})$
  - the target in n-step return
    - ✓ $G_{t:t+n} \doteq R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n V_{t+n-1}(S_{t+n})$
  - value function learning algorithm
    - ✓ $V_{t+n}(S_t) \doteq V_{t+n-1}(S_t) + \alpha \left[G_{t:t+n} - V_{t+n-1}(S_t) \right]$

Temporal difference

- n-step TD
  - 可以证明the target的期望与原value function到 v_π 的距离满足
    - ✓ $$\max_s \left| \mathbb{E}_\pi[G_{t:t+n} | S_t = s] - v_\pi(s) \right| \leq \gamma^n \max_s \left| V_{t+n-1}(s) - v_\pi(s) \right|$$
  - 可以保证n-step TD方法在恰当的近似条件下可以收敛到正确的value function

Temporal difference

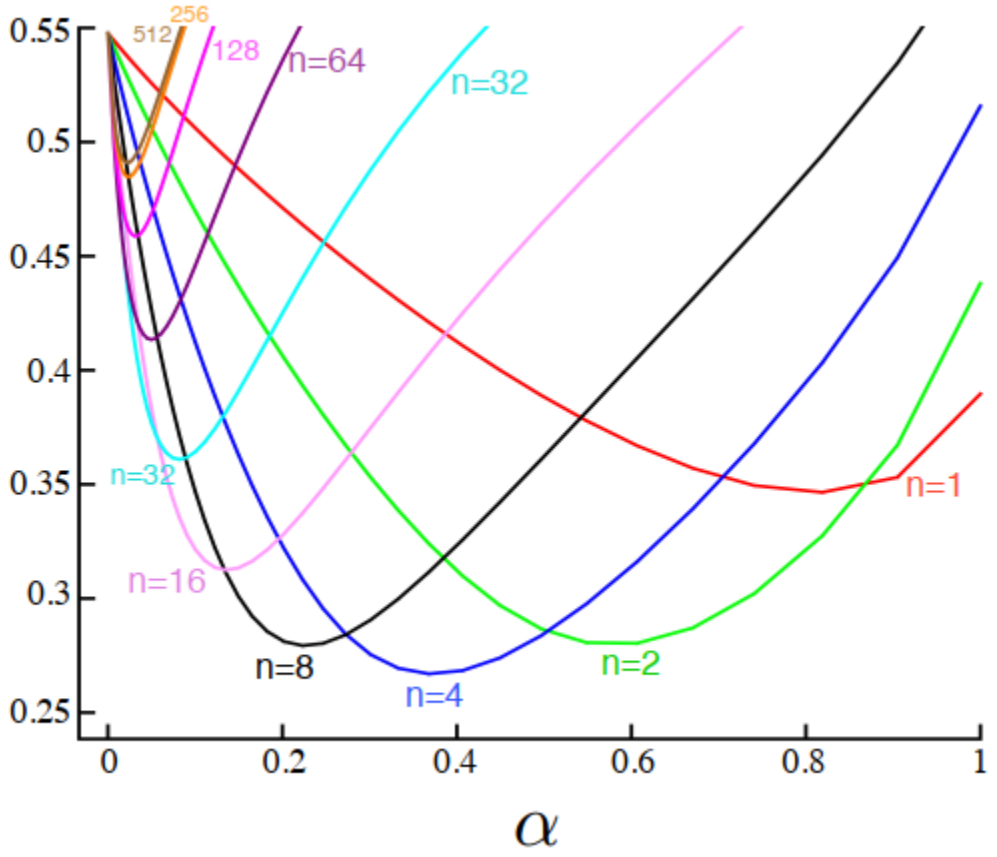
- 例子：Random Walk

A *Markov reward process*, or MRP, is a Markov decision process without actions. We will often use MRPs when focusing on the prediction problem, in which there is no need to distinguish the dynamics due to the environment from those due to the agent. In this MRP, all episodes start in the center state, C, then proceed either left or right by one state on each step, with equal probability. Episodes terminate either on the extreme left or the extreme right. When an episode terminates on the right, a reward of +1 occurs; all other rewards are zero. For example, a typical episode might consist of the following state-and-reward sequence: C, 0, B, 0, C, 0, D, 0, E, 1. Because this task is undiscounted, the true value of each state is the probability of terminating on the right if starting from that state. Thus, the true value of the center state is $v_\pi(\mathsf{C}) = 0.5$. The true values of all the states, A through E, are $\frac{1}{6}, \frac{2}{6}, \frac{3}{6}, \frac{4}{6}$, and $\frac{5}{6}$.

Temporal difference

- 例子：Random Walk

Average RMS error over 19 states and first 10 episodes



TD方法

- Sarsa
  - on-policy control method
  - value function learning algorithm
    - ✓ $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\left[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)\right]$
  - 估计 q_π，同时根据 q_π 贪心的移动策略 π
  - 所有状态动作对被无限次访问时，以概率1收敛到最优策略
  - **Sarsa (on-policy TD control) for estimating $Q \approx q_*$**

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Choose A from S using policy derived from Q (e.g., ε-greedy)
  Loop for each step of episode:
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ε-greedy)
    $Q(S, A) \leftarrow Q(S, A) + \alpha\left[R + \gamma Q(S', A') - Q(S, A)\right]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

TD方法

- Q-learning
  - off-policy control method
  - value function learning algorithm
    - ✓ $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\right]$
  - 所有状态动作对持续被更新时，以概率1收敛到最优策略

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha\left[R + \gamma \max_a Q(S', a) - Q(S, A)\right]$
    $S \leftarrow S'$
  until S is terminal

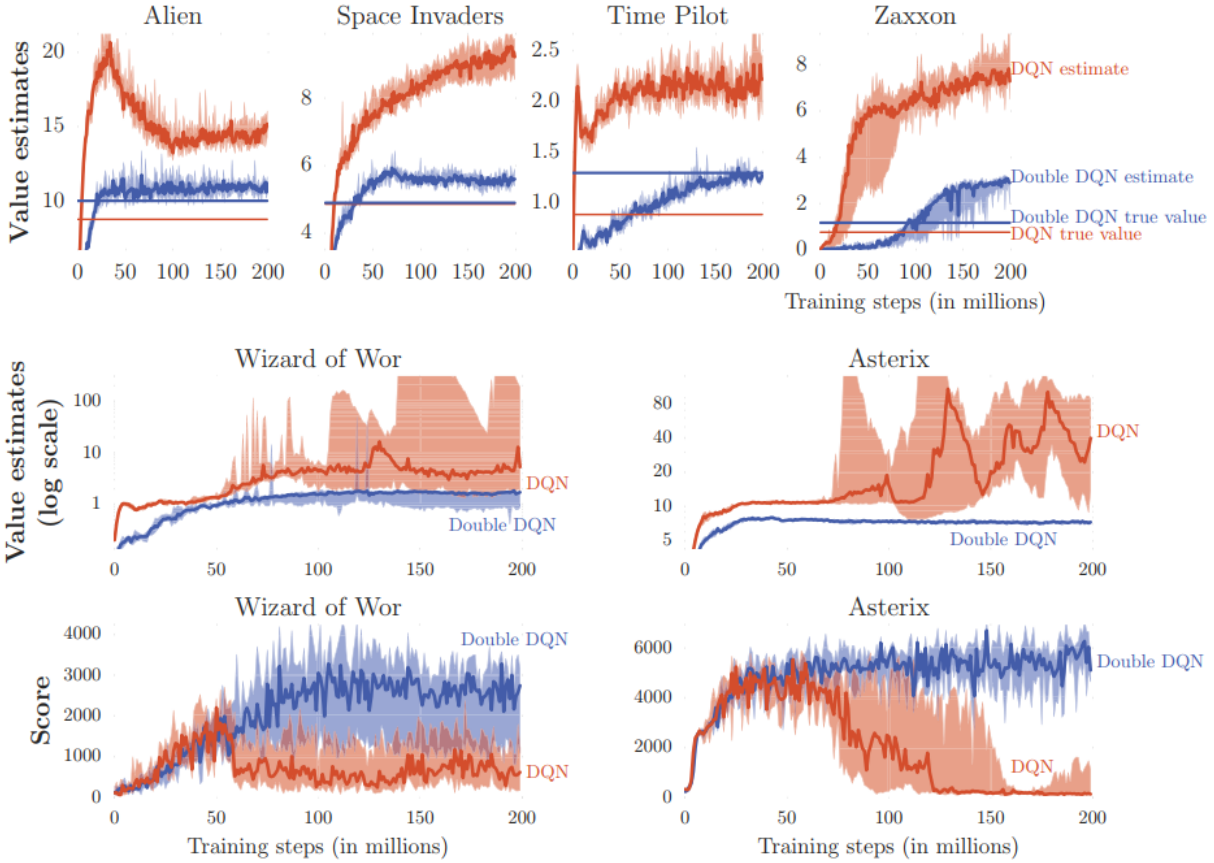
TD方法

- Sarsa与Q-learning
  - 用不同的方法计算the target
- DQN
  - value function learning algorithm
    - ✓ $\theta_{t+1} = \theta_t + \alpha(Y_t^Q - Q(s_t, a_t; \theta_t))\nabla_{\theta_t} Q(s_t, a_t; \theta_t)$
  - the target
    - ✓ $Y_t^Q = r_{t+1} + \gamma \max_a Q(s_{t+1}, a; \theta_t)$
  - 改进
    - ✓ 用target Q和Q两个网络，提高算法稳定性
    - ✓ experience replay，将样本保存到经验池再抽取，减少数据相关性
- DDQN
  - the target
    - ✓ $Y_t^{DoubleQ} = r_{t+1} + \gamma Q(s_{t+1}, \arg\max_a Q(s_{t+1}, a; \theta_t); \theta'_t)$

DQN: V Mnih, K Kavukcuoglu et al. Human-level control through deep reinforcement learning. Nature 2015
DDQN: H Van Hasselt, A Guez et al. Deep Reinforcement Learning with Double Q-Learning. AAAI. 2016

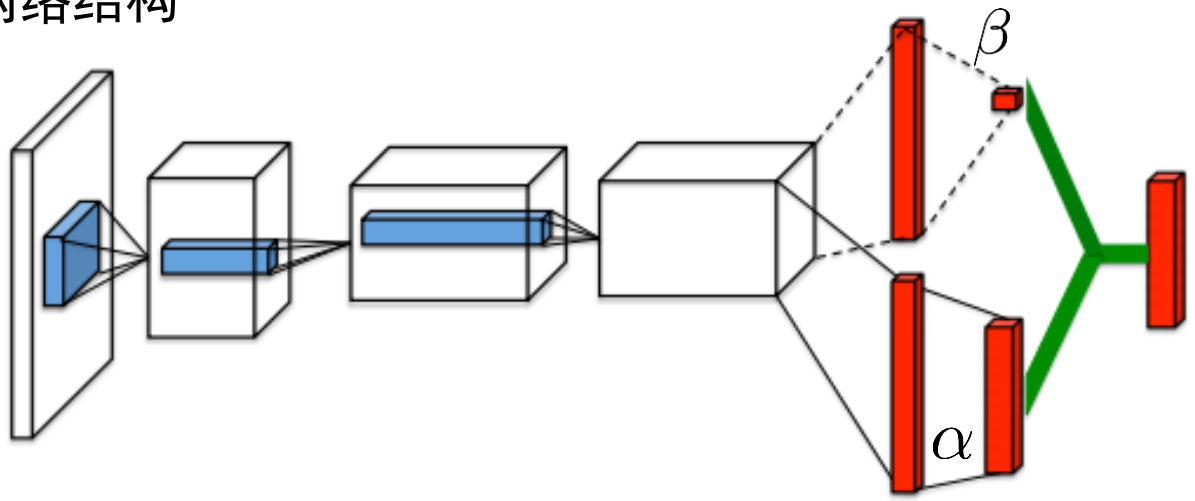
TD方法

- DDQN
  - 解释
    - ✓ Reduce overoptimism，more stable



TD方法

- Dueling Network Architectures
  - Decompose $Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + A(s,a;\theta,\alpha)$
  - 解释：
    - ✓ 更频繁地更新value stream V，不同的action共享一个V，可以更好地近似V
    - ✓ DDQN中的现象：平均Q值是15，第一与第二相差0.04；对这种影响更鲁棒
  - 网络结构

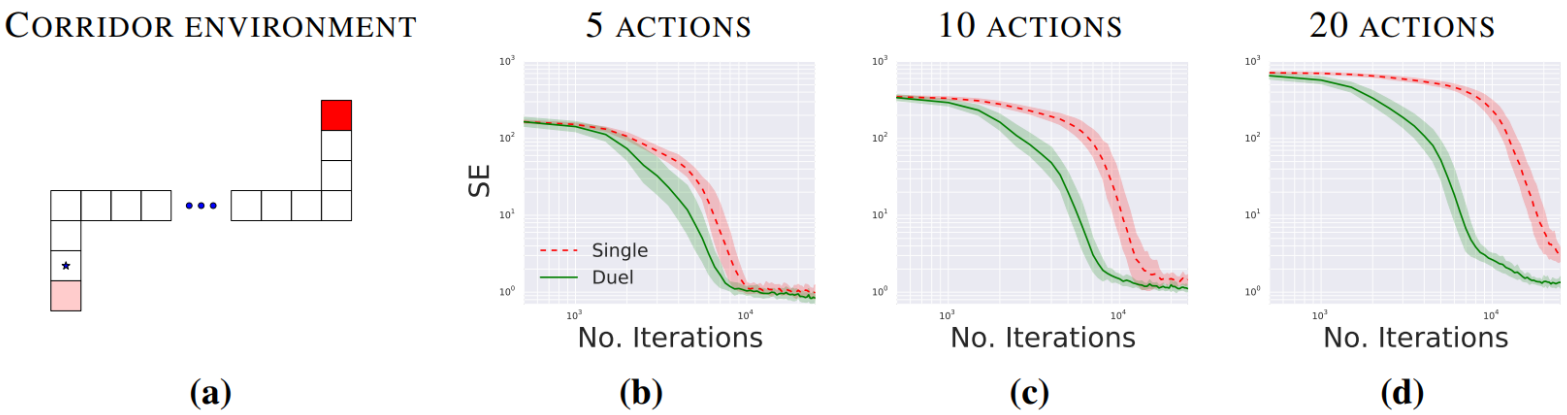


Z Wang, T Schaul et al. Dueling Network Architectures for Deep Reinforcement Learning. arXiv. 2015

TD方法

- Dueling Network Architectures
  - 实验结果

$$SE = \Sigma_{s,a}(Q(s,a;\theta) - Q^{\pi}(s,a))^2$$



Z Wang, T Schaul et al. Dueling Network Architectures for Deep Reinforcement Learning. arXiv. 2015

TD方法

- DQN + Prioritized experience replay
  - 经验池有两种改进方法，选择性保存，选择性抽取，这里研究后者
  - 用TD-Error衡量样本优先级
  - 结果：使experience replay更有效，加速学习，提高性能。

Tom Schaul, John Quan, et al. Prioritized experience replay. ICLR. 2015. DeepMind

- 简单介绍
- Temporal difference方法
- Policy gradient方法

Policy gradient

- 理论
  - 梯度方法
    - ✓ $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha\widehat{\nabla J(\boldsymbol{\theta}_t)} \quad J(\boldsymbol{\theta}) \doteq v_{\pi_{\boldsymbol{\theta}}}(s_0)$
  - 可以计算得到
    - ✓ $$\nabla J(\boldsymbol{\theta}) \propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla \pi(a|s,\boldsymbol{\theta})$$
      $$= \mathbb{E}_\pi \left[\sum_a q_\pi(S_t,a) \nabla \pi(a|S_t,\boldsymbol{\theta}) \right].$$
  - 随机梯度算法（all-actions method）
    - ✓ $$\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha \sum_a \hat{q}(S_t,a,\mathbf{w}) \nabla \pi(a|S_t,\boldsymbol{\theta})$$
    - ✓ 缺点：需要对所有动作求和

Policy gradient

- 理论
  - 随机梯度算法（一般）
    - ✓ $$\nabla J(\boldsymbol{\theta}) = \mathbb{E}_\pi\left[\sum_a \pi(a|S_t,\boldsymbol{\theta}) q_\pi(S_t,a) \frac{\nabla\pi(a|S_t,\boldsymbol{\theta})}{\pi(a|S_t,\boldsymbol{\theta})}\right]$$
      $$= \mathbb{E}_\pi\left[q_\pi(S_t,A_t)\frac{\nabla\pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})}\right]$$
      $$= \mathbb{E}_\pi\left[G_t\frac{\nabla\pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})}\right],$$
      - 用动作采样 $A_t \sim \pi$ 代替对所有动作求和
    - ✓ 一般算法 $$\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha G_t \frac{\nabla\pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})}$$
      $$= \boldsymbol{\theta}_t + \alpha G_t \nabla \ln \pi(A_t|S_t,\boldsymbol{\theta})$$
    - ✓ 改进：with baseline，减小方差
      $$\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha(G_t - b(S_t))\nabla \ln \pi(A_t|S_t,\boldsymbol{\theta})$$

Policy gradient

- Actor-Critic方法
  - 既有policy又有value function
  - 引入TD方法中的bootstrapping

$$\begin{aligned}\boldsymbol{\theta}_{t+1} &\doteq \boldsymbol{\theta}_t + \alpha(G_{t:t+1} - \hat{v}(S_t, \boldsymbol{w}))\nabla \ln \pi(A_t|S_t, \boldsymbol{\theta}) \\ &= \boldsymbol{\theta}_t + \alpha(R_{t+1} + \gamma\hat{v}(S_t + 1, \boldsymbol{w}) - \hat{v}(S_t, \boldsymbol{w}))\nabla \ln \pi(A_t|S_t, \boldsymbol{\theta}) \\ &= \boldsymbol{\theta}_t + \alpha\delta_t \nabla \ln \pi(A_t|S_t, \boldsymbol{\theta})\end{aligned}$$

- off-policy policy gradient
  - 重要性采样

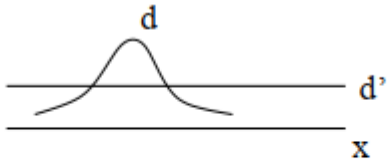


Figure 1. Different target and sampling distributions

Policy gradient

- off-policy policy gradient
  - 重要性采样

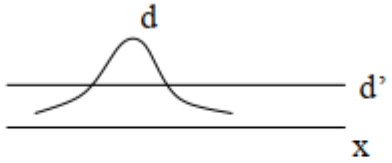


Figure 1. Different target and sampling distributions

$$E_d\{x\} = \int_x x\,d(x)\,dx = \int_x x\frac{d(x)}{d'(x)}d'(x)\,dx$$

$$= E_{d'}\left\{x\frac{d(x)}{d'(x)}\right\},$$

which leads to the importance sampling estimator,

$$\approx \frac{1}{n}\sum_{i=1}^{n} x_i\frac{d(x_i)}{d'(x_i)}$$

Policy gradient

- off-policy policy gradient
  - $$\nabla_\theta J_\beta(\pi_\theta) \approx \int_{\mathcal{S}} \int_{\mathcal{A}} \rho^\beta(s) \nabla_\theta \pi_\theta(a|s) Q^\pi(s,a) \mathrm{d}a\mathrm{d}s$$
    $$= \mathbb{E}_{s\sim\rho^\beta, a\sim\beta}\left[\frac{\pi_\theta(a|s)}{\beta_\theta(a|s)} \nabla_\theta \log \pi_\theta(a|s) Q^\pi(s,a)\right]$$

- off-policy policy evaluation algorithm(in control problem)

$$RQ(x,a) := Q(x,a) + \mathbb{E}_\mu\left[\Sigma_{t\geq 0}\left(\Pi_{s=1}^{t} c_s\right)\left(r_t + \gamma \mathbb{E}_\pi Q\left(x_{t+1}, \cdot\right) - Q\left(x_t, a_t\right)\right)\right]$$

  - c_s — traces of the operator. non-negative coefficients
  - Importance sampling
    - ✓ $c_s = \frac{\pi(a_s|x_s)}{\mu(a_s|x_s)}$
    - ✓ large variance, due to the product

Policy gradient

- off-policy policy evaluation algorithm(in control problem)

$$RQ(x,a) := Q(x,a) + \mathbb{E}_{\mu}\left[\Sigma_{t\geq 0}\left(\Pi_{s=1}^{t} c_s\right)\left(r_t + \gamma \mathbb{E}_{\pi} Q\left(x_{t+1}, \cdot\right) - Q\left(x_t, a_t\right)\right)\right]$$

  - off-policy $Q^{\pi}(\lambda)$ and $Q^{*}(\lambda)$
    - ✓ $c_s = \lambda$
    - ✓ contraction mapping around $Q^{\pi}(\lambda)$ for $\lambda < \frac{1-\gamma}{\gamma^{\varepsilon}}$, not safe
  - Tree-backup, TB(λ)
    - ✓ $c_s = \lambda \pi(a_s|x_s)$
    - ✓ near on-policy: cut the traces c_s, not efficient
  - Retrace(λ)
    - ✓ $c_s = \min\left(1, \frac{\pi(a_s|x_s)}{\mu(a_s|x_s)}\right)$
    - ✓ take the best of the three previous algorithms
- truncated importance sampling with bias correction

Retrace: Remi Munos, Thomas Stepleton et al. Safe and efficient off-policy reinforcement learning. NIPS. 2016

Policy gradient

- 算法
  - A3C
  - DDPG
  - ACER

谢谢